

REMARKS

Claims 1-22 are currently pending in the Application. In an office action dated September 25, 2003 ("Office Action"), the Examiner rejected claims 1-22 under 35 U.S.C. § 103(a) as being unpatentable over Liu et al., U.S. Patent No. 6,009,541 ("Liu") in view of Shankman et al., U.S. Patent No. 6,381,656 ("Shankman"). Applicant's representative respectfully traverses these 35 U.S.C. § 103(a) rejections.

Consider claim 1, provided below for the Examiner's convenience:

1. A testing platform within a computing resource environment, the testing platform comprising:
 - a test execution engine that receives input commands and initiates processing of the input commands;
 - a test routine; and
 - components that serve to adapt hardware and software interfaces of the computing resource environment to the test execution engine and test routine and that shield the test execution engine and test routine from dependencies on the hardware and software interfaces of the computing resource environment, including the operating system interface. (emphasis added)*

Please note that the claimed testing platform includes components that shield the test execution engine and test routine from dependencies on the hardware and software interfaces of the computing resource environment, including the operating system interface. Again, for the Examiner's convenience, the following paragraph from the current application, beginning on line 22 of page 5, with reference to Figure 2 of the current application, is provided to further understanding of claim 1:

The testing platform includes a core test execution engine 212 that includes a central execution loop that continuously executes in order to run one or more test routines according to various user-input and programmed parameters. Both the test routine 210 and the test execution engine 212 are shielded by additional testing platform components from dependencies on the computing resources and external entities 202, 204-205, 206-207, and 208 within the environment in which the testing platform runs. The test execution engine 212 interfaces to data output and data storage devices 204-205 via a result handler component 214. The test execution engine interfaces to operating-system-provided functionality 202 via various components diagrammed together in Figure 2 as a generalized operating system adaptor 216. The test execution engine interfaces to user I/O devices 206-207 via a user I/O component 218. The test execution engine interacts with the test routine via a mode component 220, a sequencer component 222, and a test executor component 224. The test routine 210 interfaces with the result handler component via a first test link component 226 and interfaces with the operating system adaptor, user I/O handler, and a communications interface 228 via a

second test link component 230. The communications interface component 228 serves to interface the test routine 210 with a hardware component 208 tested by the test routine.

In both the quoted text, and in Figure 2, it is abundantly clear that Applicant's test platform is a self-contained, logically isolated group of components external to, and apart from, the operating system of the computing resource environment in which the testing platform runs. Applicant deliberately designed the claimed testing platform in this way for, among others, portability reasons, as detailed in the below quoted portion of the paragraph beginning on line 21 of page 8 of the current application:

The highly modular and onion-like layers of shielding provided by the functional components of the testing platform allow for the high levels of enhancability, adaptability, and portability of both the testing platform and of test routines developed to test various hardware and software components. The testing platform, for example, can be ported to almost any computing resource environment, including to many different operating systems and computer platforms, without the need to modify the internal execution loop within the test execution engine, nor functional components such as the test sequencer.

Next, please again consider the cited references. Liu's extended BIOS testing diagnostic routine "is a separate block of code stored within a common ROM with the BIOS test routine. In another embodiment, the extended diagnostic routine is stored in a hard disk drive" (column 3, lines 9-13). In one embodiment, "the diagnostic routine begins by reading to and writing from the central processing unit (CPU). The routine may check several standard CPU registers which record errors ..." (column 3, line 67 – column 4, line 3). In one embodiment, "the diagnostic routine also performs a test of the Peripheral Component Interconnect (PCI) cards and buses or other type of bus components (column 4, lines 6-8). Therefore, as also noted by the Examiner, Liu most explicitly and decidedly does not include "components that serve to adapt hardware and software interfaces of the computing resource environment to the test execution engine and test routine and that shield the test execution engine and test routine from dependencies on the hardware and software interfaces of the computing resource environment, including the operating system interface," as clearly claimed in claim 1. Liu teaches exactly an opposite strategy, with Liu's diagnostic routine – an simple, unshielded code block – directly accessing hardware components such as CPU registers, PCI cards, and buses. In Applicant's representative's viewpoint, Liu is completely

unrelated to the claimed invention. Indeed, Applicant's testing platform can be furnished with a test routine that test particular hardware components, but Applicant's test routine interfaces to layers of tiered, hierarchical shielding components, rather than directly to CPU registers, buses, and other computer resources within the computer-resource environment in which Applicant's testing platform happens to be running.

Shankman appears to be equally unrelated to the current, claimed invention. Shankman provides a method and system for monitoring performance of I/O processors included within server hardware by introducing a virtual monitor diagrammed in Figure 4 into each I/O processor, and including an I/O monitor, diagrammed in Figure 5, within the server to collect I/O messages from the virtual monitors, and reporting performance data to an I/O user interface (Abstract of Shankman, Figure 4, and Figure 5). As stated in the Summary, Shankman's virtual-monitor-and-I/O-monitor system "monitors message traffic passing through the input/output ("I/O") processors within servers in a computing system in order to analyze the I/O processing within the computing system as a whole" (column 3, lines 1-5). *Shankman's virtual-monitor-and-I/O-monitor system is not a testing platform.* The concept of testing is well known in computer science, as stated in the current application beginning on line 19 of page 1: "A sizable body of theory and technology related to testing and quality assurance has evolved along with the evolution of computers and software technologies." Testing involves exhaustive exercise of, or at least well-planned sampling of, the state spaces involved in running software programs and hardware devices. For example, a software routine is commonly tested by invoking the routine with as many different parameter values and input data as possible, to insure that each instruction within the routine executes properly for the range of input values that may be encountered. By contrast, performance monitoring is a different area of computing. In performance monitoring, statistics and data related to operation of a routine, device, or system are collected and analyzed to ascertain how the routine, device, or system operates. Performance monitoring may well be employed in testing, but is not itself testing. For example, a test routine could iteratively alter various system states and then monitor the resulting performance. But testing involves, in such cases, experimentation – developing a hypothesis, perturbing a system, observing the effects of the perturbation, refining the

hypothesis, and continuing iteratively until a hypothesis is validated by observation. Performance monitoring may comprise the observational step in an experimental procedure, but is, by itself, neither a test nor an experiment.

Shankman, for example, neither discloses nor suggests a test routine, or, in other words, a routine that implements an experimental test procedure, since performance monitoring involves merely passively collecting performance data. Applicant's test platform "includes a core test execution engine 212 that includes a central execution loop that continuously executes in order to run one or more test routines according to various user-input and programmed parameters." Shankman, for this reason alone, neither teaches nor suggests the testing platform of claim 1.

Not only is Shankman's system not a testing platform, Shankman's virtual-monitor-and-I/O-monitor system is also not shielded from dependencies on the hardware and software interfaces of the computing resource environment, including the operating system interface. Consider, for example, Figure 5 of Shankman and the accompanying text beginning at line 3 of column 8:

FIG. 5 is a block diagram depicting the I/O monitor 201 shown in FIG. 2 and its interfaces to other elements of the monitoring system, according to an embodiment of the invention. The I/O monitor interfaces with the I/O user interface 220 running on the workstation 200, the virtual adapters 204-206 and with server devices 501 on the server 101. The I/O monitor 201 interfaces with the server devices 501, including both I₂O devices 502 and non-I₂O devices 503.

As shown in Figure 5, and explicitly stated in the above text, Shankman's I/O-monitor directly interacts with hardware devices. Moreover, Shankman's virtual monitor directly interacts with the operating system of the I/O processor in which it runs, as clearly shown by the double arrow interconnecting the communications module of the virtual monitor 402 and the I/O processor operating system 430 in Figure 4. There is no shielding of Shankman's virtual-monitor-and-I/O-monitor system from hardware dependencies – instead, components of Shankman's virtual-monitor-and-I/O-monitor system are directly embedded into the computer system components and directly interface with them.

The Examiner points to column 1, lines 6-10 of Shankman to show that Shankman provides a testing platform, but, in Applicant's representative's opinion, the Examiner has misstated what those lines of Shankman clearly state. The cited passage

reads, as follows: "The invention relates to a computer input/output ("I/O") subsystem monitor for testing, debugging and performance analysis and, more particularly, to I/O subsystem monitoring in computing systems using an intelligent I/O ("I₂O") architecture." (emphasis added) Certainly performance monitoring may be related to testing, as discussed above, but it is not testing, and Shankman does not disclose a testing platform. The Examiner states that, in column 4, lines 15-13, Shankman discloses adapters that conform to an intelligent I/I architecture that adapts the monitoring execution system to the operating system, but Shankman does not disclose anything of the kind in these cited lines. Instead, Shankman describes why he describes the components of his system that execute on I/O processors as "virtual adapters." Shankman states: "A virtual adapter inserts itself into the message stream of an I/O processor ("IOP") and may then monitor I2O messages passing through the IOP. An 'adapter' in I2O parlance comprises any hardware entity compliant with the I2O architecture." The term "operating system" does not appear in the cited lines.

In summary, Liu's BIOS diagnostic testing routine is a standard testing routine that intimately and directly interacts with the hardware components of a computer system. Shankman discloses a performance monitoring system that also intimately interacts with both hardware components and operating systems. Neither reference discloses, teaches, or suggests "a test execution engine that receives input commands and initiates processing of the input commands" and neither reference discloses, teaches, or suggests "components that serve to adapt hardware and software interfaces of the computing resource environment to the test execution engine and test routine and that shield the test execution engine and test routine from dependencies on the hardware and software interfaces of the computing resource environment, including the operating system interface." Liu does disclose a testing routine. Shankman fails even to disclose a testing routine, and therefore is unrelated to any of the three elements of claim 1.

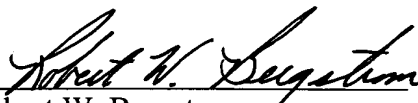
Applicant's representative would like to emphasize that the disclosed and claimed testing platform and test routines represent a significant departure from commonly employed testing applications and systems. Applicant's testing platform is truly portable, because, being shielded from any particular computer resource environment. Applicant's test routines are also shielded from any particular computer resource environment. Applicant's representative worked for 13 years in the computer



industry, and has been involved for more than 8 years in patent prosecution of computer-related inventions, and has not professionally encountered a similar testing architecture.

All of the claims remaining in the application are now clearly allowable. Favorable consideration and a Notice of Allowance are earnestly solicited.

Respectfully submitted,
John S. Eden
Olympic Patent Works PLLC


Robert W. Bergstrom
Registration No. 39,906

Enclosures:
Postcards (2)
Transmittal in duplicate

Olympic Patent Works PLLC
P.O. Box 4277
Seattle, WA 98194-0277
206.621.1933 telephone
206.621.5302 fax